Tree Adjoining Grammars in Noncommutative Linear Logic

— Extended Abstract —

V. Michele Abrusci[1], Christophe Fouqueré[2], and Jacqueline Vauzeilles[2]

[1] CILA, Universitá di Bari, 70121 Bari, Italy
[2] LIPN-CNRS URA 1507, Université Paris-Nord, 93430 Villetaneuse

Abstract. This paper[1] presents a logical formalization of Tree-Adjoining Grammar (TAG). TAG deals with lexicalized trees and two operations are available: substitution and adjunction. Adjunction is generally presented as an insertion of a tree inside another, surrounding the subtree at the adjunction node. This seems to be contradictory with standard logical ability. We prove that some logic, namely a fragment of non-commutative intuitionistic linear logic (N-ILL), can serve this purpose. Briefly speaking, linear logic is a logic considering facts as *resources*. N-ILL can then be considered either as an extension of Lambek calculus, or as a restriction of linear logic. We model the TAG formalism in four steps: trees (initial or derived) and the way they are constituted, the operations (substitution and adjunction), and the elementary trees, i.e. the grammar. The sequent calculus is a restriction of the standard sequent calculus for N-ILL. Trees (initial or derived) are then obtained as the closure of the calculus under two rules that mimic the grammatical ones. We then prove the equivalence between the language generated by a TAG grammar and the closure under substitution and adjunction of its logical representation. Besides this nice property, we relate parse trees to logical proofs, and to their geometric representation: *proofnets*. We briefly present them and give examples of parse trees as proofnets. This process can be interpreted as an assembling of blocks (proofnets corresponding to elementary trees of the grammar).

1 Introduction

This paper presents a logical formalization of Tree-Adjoining Grammar (TAG, [8], [9]). TAG deals with lexicalized trees and two operations are available: substitution and adjunction. A set of (elementary) trees is associated to each lexical item. TAG is a tree rewriting system: the parsing process consists in applying operations to trees in order to obtain a (derived) tree whose sequence of leaves is a sentence. Adjunction increases the expressive power of the formalism in such a way that non context-free languages can be represented although the parsing process is done in a polynomial time. Adjunction is generally presented as

[1] An extended version is currently in submission and available as a technical report [5]

an insertion of a tree inside another, surrounding the subtree at the adjunction node. This seems to be contradictory with standard logical ability. We prove hereafter that some logic, namely a fragment of non-commutative intuitionistic linear logic (N-ILL, [2,3]), can serve this purpose. Briefly speaking, linear logic, developed by Girard [6], is a logic considering facts as *resources*. N-ILL can then be considered either as an extension of Lambek calculus, or as a restriction of linear logic. Nevertheless, viewing logical literals as resources allows a straight and natural mapping from a derivational formalism (such as TAG) to such a logic: there is no need for indexing pieces of trees or words of a sentence. Since we are not interested in disjunction in this paper, we only need the intuitionistic part of linear logic. Finally, non-commutativity is necessary insofar as we aspire to take care of word order.

We model the TAG formalism in four steps: trees (initial or derived) and the way they are constituted, the operations (substitution and adjunction), and the elementary trees, i.e. the grammar. Labels occurring in the grammar constitute the set of propositional variables we need. The sequent calculus is a restriction of the standard sequent calculus for N-ILL: there are identity axioms ($A \vdash A$) and rules for introducing connectives ($\otimes$ at left hand side, $\multimapinv$ at right hand side). $\multimapinv$ is the left implication, $\otimes$ is one of the two 'and' connectives available in Linear Logic and its variants. We prove that this restricted calculus is closed under two rules that mimic the grammatical operations. Trees (initial or derived) are then obtained as the closure of the calculus under these two rules. In fact, trees are represented as (provable) sequents in an almost classical way. The right hand side is the variable labeling the mother node of the tree. The left hand side is a sequence of formulas of the following kinds: A for some leaf A of the tree, $A \multimapinv B_1 \otimes \cdots \otimes B_n$ where A is the label of some internal node and B_i are the labels of its daughters, $A \multimapinv A$ whenever A is a node where an adjunction can take place. This latter kind of formula can be grammatically interpreted as if such an A was split into two nodes with the same label, linked by some "soft" relation. The set of elementary trees of a TAG grammar $\mathcal{G}'$ is then represented as a subset M of the sequents in the closure of the calculus under the two previous rules. We then prove the equivalence between the language generated in TAG by such a grammar $\mathcal{G}'$ and the closure under substitution and adjunction of the logical representation M. Note that our interpretation of adjunction is very close to the use of *quasi-trees* described in [13].

Besides this nice property, we relate parse trees to logical proofs, and to their geometric representation: *proofnets*. As for linear logic, there exists in N-ILL a correspondence between proofs and some sort of nets, called proofnets. We briefly present proofnets and give examples of parse trees as proofnets. This enables a new point of view on the parsing process. This process can be interpreted as an assembling of blocks (proofnets corresponding to elementary trees of the grammar), and also as a circulation of information through links relating nodes of the proofnets.

The paper is organized in four parts. Section 2 describes the TAG formalism. We recall the terminology and show how substitution and adjunction operate on trees. Section 3 gives a survey of noncommutative linear logic and relates it to Lambek calculus and linear logic. We propose in section 4 a logical formulation of TAG in a fragment of N-ILL, and prove the correspondence between the two. Section 5 is devoted to the representation of proofs as proofnets.

2 Tree Adjoining Grammars

The Tree Adjoining Grammar formalism (TAG) is a tree generating formalism introduced in [8], linguistically motivated (see for example, [1,10]), and with formal properties studied in [14–16]. A TAG is defined by two finite sets of trees composed by means of the substitution and adjunction operations[2].

Definition 1. A TAG G is a 5-uple (V_N, V_T, S, I, A) where

- V_N is a finite set of non-terminal symbols,
- V_T is a finite set of terminal symbols,
- S is a distinguished non-terminal symbol, the *start* symbol,
- I is a set of *initial* trees,
- A is a set of *auxiliary* trees.

An *elementary* tree is either an initial tree or an auxiliary tree. Both *initial* and *auxiliary* trees are trees with at least one leaf labeled by a terminal node (the grammar is a so-called *lexicalized* one). An *auxiliary* tree must furthermore have a leaf (the *foot* node, marked with a star $\star$) with the same label as the root node. Each non-terminal node is marked as adjoinable or non-adjoinable (in this case, the node is marked NA). Each internal node must obviously be labeled by a non-terminal node.[3] A *derived* tree is either an initial tree or a tree obtained from derived trees by means of the two available operations.

In conformity with the literature, we will use α to refer to an initial tree, β to refer to an auxiliary tree, γ to refer to some derived tree. Examples of initial and auxiliary trees are given in fig. 1. Two TAGs are defined: $G_1 = (\{S\}, \{a, b, c, d, \epsilon\}, S, \{\alpha_1\}, \{\beta_1\})$ (ϵ is the empty word) and $G_2 = (\{S, VP, NP, N\}, \{the, man, walks\}, S, \{\alpha_2, \alpha_3, \alpha_4\}, \emptyset)$.

The *substitution* operation is defined as usual. A non-terminal leaf of a tree may be expanded with a tree whose root node has the same label. Leaves that accept substitution are marked with a down arrow $\downarrow$. The *adjunction* operation is a little bit more complicated. It supposes a derived tree with a non-terminal

[2] Originally, there was no need for a substitution operation as initial trees were rooted at S, thus labeling a sentence. We refer here to the Lexicalized-TAG formalism where this constraint disappears on behalf of the substitution operation. However, we maintain the name TAG.

[3] In some versions, non-terminal nodes of elementary trees are labeled by a *set* of (auxiliary) trees that can be adjoined at this node. In the case of an empty set, the node is obviously non-adjoinable. For the sake of clarity, we simplify the definition to only take into account the *boolean* adjoinable property: either the node is adjoinable or it is non-adjoinable (NA).

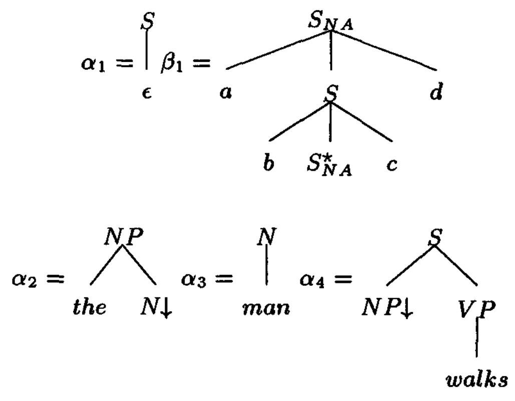
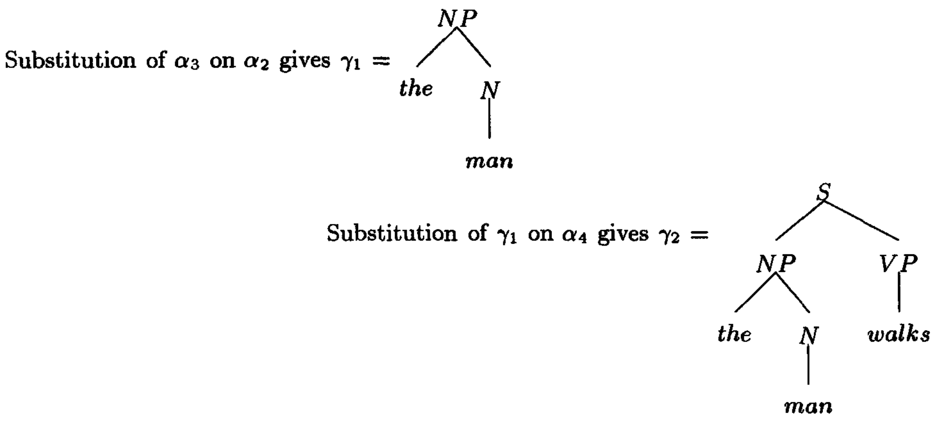


Fig. 1. Elementary trees.

node, say X, possibly internal and not marked NA, and an auxiliary tree with root node X. The operation consists in:

- excising the subtree with root labeled X in the derived tree,
- inserting the auxiliary tree at node labeled X in the derived tree,
- finally, inserting the excised subtree at the foot node (labeled X and marked with a star $\star$) in the auxiliary tree.

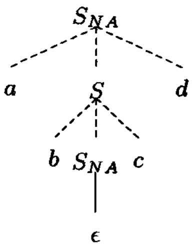


Substitution of α_3 on α_2 gives $\gamma_1 =$
NP
the
N
man
Substitution of γ_1 on α_4 gives $\gamma_2 =$
S
NP
VP
the
N
walks
man

Fig. 2. Substitution results

Examples of these operations are given in fig. 2 and 3. In order to clarify the way adjunction is done, the adjoined tree β_1 has its links dashed in the derived trees γ_3 and γ_4. Obviously, there is only one kind of link. We write $\gamma_1 \Rightarrow_G \gamma_2$ when γ_2 is the result of an adjunction or a substitution of an elementary tree

Adjunction of β_1 on α_1 gives $\gamma_3 =$



Adjunction of β_1 on γ_3 gives $\gamma_4 =$

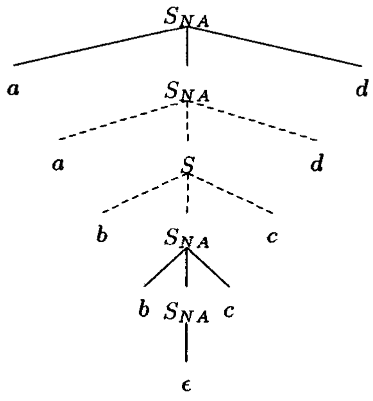


Fig. 3. Adjunction results

of a TAG G on the derived tree γ_1. $\Rightarrow_G^*$ is the reflexive, transitive closure of $\Rightarrow_G$. The set $\{\gamma / \exists \alpha \in G$ and $\alpha \Rightarrow_G^* \gamma\}$ is noted $T(G)$. The language $L(G)$ generated by a TAG G is the set of strings, i.e. sequences of leaves of trees in $T(G)$ when the leaves of these trees are only labeled with terminal nodes, and whose root is the start symbol. Hence, $L(G_1) = \{a^n b^n c^n d^n / n \geq 0\}$ and $L(G_2) = \{\textit{the man walks}\}$.

3 Noncommutative Intuitionistic Linear Logic

Linear logic was introduced by Girard [6] as a "resource conscious logic". In other words, while classical logic deals with static descriptions, linear logic considers propositions as finite resources. Hence, 'A' and '*A and A*' are equivalent in classical logic, this is (generally) not the case in linear logic. The easiest technical way to investigate this difference is to consider the Gentzen sequent calculus for these logics. A *sequent* is of the form $\Gamma \vdash \Delta$ where Γ and Δ stand for sequences of well-formed formulae w.r.t. the language of the logic. It expresses the fact that the (multiplicative) disjunction of formulas in Δ is a consequence of the (multiplicative) conjunction of formulas in Γ. A *sequent calculus* is a set of rules specifying the provable sequents, being given a set of axioms. A *proof* of a sequent is then the successive application of sequent rules beginning with axioms, i.e. a tree with the proved sequent as the root of the tree (at the bottom) and whose leaves are axioms (at the top). Rules of a sequent calculus introduce in some way connectives in the right or left hand side of a sequent. Hence, in the case of a simple calculus, sequent calculus is constructive in the sense that, given a sequent to be proved, there exists a constructive way to find the proof. We refer the reader to [12] for a survey on various systems of deduction and the relations between sequent calculus, Hilbert-style systems and natural deduction. Besides these introduction rules and axioms, we find structural rules that govern the structure of a sequent. In classical logic, the set of structural rules consists in weakening, contraction and exchange (cf fig. 4 where A, B are formulas, $\Gamma, \Gamma', \Delta, \Delta'$ are sequences of formulas). Weakening and contraction allow the arbitrary copy of formulas: having a formula A as a hypothesis or conclusion is equivalent to having it twice (or more). This point of view contradicts the notion of resource, hence these two structural rules are omitted in linear logic. However special connectives, namely the *exponentials* of-course '!' and why-not '?' have these properties. The exchange rule is responsible for commutativity of the comma (in the right side and in the left side): the order of hypotheses or conclusions does not matter. This rule is no longer valid in the noncommutative version of linear logic.

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\ (l-weakening) \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\ (r-weakening)$$

$$\frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta}\ (l-contraction) \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ (r-contraction)$$

$$\frac{\Gamma, B, A, \Gamma' \vdash \Delta}{\Gamma, A, B, \Gamma' \vdash \Delta}\ (l-exchange) \qquad \frac{\Gamma \vdash \Delta, B, A, \Delta'}{\Gamma \vdash \Delta, A, B, \Delta'}\ (r-exchange)$$

Fig. 4. Structural rules

However, and this is already true in linear logic, the logical interpretation of 'and' and 'or' is not as simple as it is in classical logic. We need to distinguish two 'and' ($\otimes$ 'times', & 'with') and two 'or' ($\invamp$ 'par', $\oplus$ 'plus'), hence inducing four constants: $1, \top, \bot, 0$ (respective neutral elements for the previous connectives). In fact, connectives are related in such a way that they form two groups: the multiplicative group $(\otimes, \invamp, 1, \bot)$ and the additive group $(\&, \oplus, \top, 0)$. We only use hereafter the multiplicative group. There are obviously fundamental reasons for this proliferation but these explanations are outside the scope of this paper. The negation and the implication are however of special interest. In (commutative) linear logic, there are only one negation $\cdot^{\perp}$ and one (linear) implication $\multimap$. In the noncommutative case, they have also to be split: there are a pre- ${}^{\perp}\cdot$ and a post- negation $\cdot^{\perp}$ and a pre- $\multimapinv$ and a post- implication $\multimap$. These two implications have to be related with the two operations in Lambek calculus: $\multimap$ with $\backslash$ and $\multimapinv$ with $/$. The implications may be defined in the following way:

$B \multimap\!\!\!- A \equiv B \parr^{\perp} A$ and $A \multimap B \equiv A^{\perp} \parr B$. We give in fig. 5 the one-sided sequent calculus for the multiplicative fragment of noncommutative linear logic N-LL, and in fig. 6 the two-sided sequent calculus for the multiplicative fragment of intuitionistic noncommutative linear logic N-ILL: sequent calculus for N-LL and sequent calculus for N-ILL satisfy the cut-elimination theorem[4]; however we make use of cut-rules in section 4. Note that if $\Gamma \vdash A$ is provable in the multiplicative intuitionistic noncommutative linear logic, then $\vdash (\Gamma^*)^{\perp}, A^*$ is provable in the multiplicative noncommutative linear logic, where:

- for each formula A of intuitionistic noncommutative linear logic, A^* is a formula of noncommutative linear logic defined as follows
  - $p^* = p$, for every propositional letter p
  - $(B \otimes C)^* = B^* \otimes C^*$, $(B \multimap C)^* = (B^*)^{\perp} \parr C^*$, $(B \multimap\!\!\!- C)^* = B^* \parr^{\perp}(C^*)$
- for each finite sequence $A_1, \ldots, A_n$ of formulas of intuitionistic noncommutative linear logic, $(A_1, \ldots, A_n)^* = (A_1)^*, \ldots, (A_n)^*$
- for each finite sequence $A_1, \ldots, A_n$ of formulas of noncommutative linear logic, $(A_1, \ldots, A_n)^{\perp} = (A_n)^{\perp}, \ldots, (A_1)^{\perp}$

We give hereafter two examples of proofs in order to show the way the sequent calculus can be used. We choose logical translations of a Lambek grammar to pinpoint the obvious relation with Lambek formalism. The first one (fig. 7) is a straightforward translation in a Lambek style, being given the two implications. The second proof (fig. 8) interprets the Lambek grammar in a derivation-style, we only need one implication and the connective times. The proofs use cuts: they can be withdrawn as the logic enjoys the cut-elimination theorem, but we think it can help understanding the process. Moreover, we associate to each lexical item a (provable) sequent we use as a proper axiom (label 'lex'). The following sections include other examples and emphasize the usefulness of noncommutative linear logic in the linguistic domain. The lexicon in a Lambek-style is the following one:

John	: NP
gives	: ((NP\ S)/NP)/NP
Mary	: NP
a	: NP/N
book	: N

4 The calculus $\mathcal{A}$ (a fragment of N-ILL)

The formalization of TAG in N-ILL relies mainly on a logical presentation of the two operations substitution and adjunction together with a correspondence between proofs and trees. As already shown in the previous section, the substitution operation is nothing else but the application of some sort of cut-rule we call the *atomic cut-rule*. Interpreting the adjunction operation is really the main difficulty. The adjunction results from two atomic cut-rules between the sequent corresponding to the adjunction tree and two suitable sequents corresponding

[4] For each proof, there exists a cut-free proof with the same conclusion.

Alphabet:

- propositional letters: $a, b, c, \ldots$
- for each propositional letter p and each integer $n > 0$

$$p^{\overbrace{\perp\cdots\perp}^{n \text{ times}}} \text{ and } {}^{\overbrace{\perp\cdots\perp}^{n \text{ times}}}p$$

- connectives: $\otimes, \parr$

Formulas: usual definition
Sequents: $\vdash \Gamma$ where Γ is a finite sequence of formulas
Metalinguistic definition of $A^{\perp}$ and ${}^{\perp}A$ s.t. ${}^{\perp}(A^{\perp}) = ({}^{\perp}A)^{\perp} = A$, for every formula A:

$$(p^{\overbrace{\perp\cdots\perp}^{n \text{ times}}})^{\perp} = p^{\overbrace{\perp\cdots\perp}^{n+1 \text{ times}}} \qquad ({}^{\overbrace{\perp\cdots\perp}^{n \text{ times}}}p)^{\perp} = {}^{\overbrace{\perp\cdots\perp}^{n-1 \text{ times}}}p$$

$${}^{\perp}(p^{\overbrace{\perp\cdots\perp}^{n \text{ times}}}) = p^{\overbrace{\perp\cdots\perp}^{n-1 \text{ times}}} \qquad {}^{\perp}({}^{\overbrace{\perp\cdots\perp}^{n \text{ times}}}p) = {}^{\overbrace{\perp\cdots\perp}^{n+1 \text{ times}}}p$$

$$(B \otimes C)^{\perp} = C^{\perp} \parr B^{\perp} \qquad (B \parr C)^{\perp} = C^{\perp} \otimes B^{\perp}$$

$${}^{\perp}(B \otimes C) = {}^{\perp}C \parr {}^{\perp}B \qquad {}^{\perp}(B \parr C) = {}^{\perp}C \otimes {}^{\perp}B$$

Rules of sequent calculus:

$$\frac{}{\vdash A^{\perp}, A}\ (axiom) \qquad \frac{\vdash \Gamma_1, A, \Gamma_2 \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma_1, \Delta, \Gamma_2}\ (cut_1) \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta_1, A^{\perp}, \Delta_2}{\vdash \Delta_1, \Gamma, \Delta_2}\ (cut_2)$$

$$\frac{\vdash \Delta_1, A, B, \Delta_2}{\vdash \Delta_1, A \parr B, \Delta_2}\ (r\,\parr) \qquad \frac{\vdash \Gamma_1, A, \Gamma_2 \quad \vdash B, \Delta}{\vdash \Gamma_1, A \otimes B, \Delta, \Gamma_2}\ (r_1\otimes) \qquad \frac{\vdash \Gamma, A \quad \vdash \Delta_1, B, \Delta_2}{\vdash \Delta_1, \Gamma, A \otimes B, \Delta_2}\ (r_2\otimes)$$

Fig. 5. Language and sequent calculus for Multiplicative Noncommutative Linear Logic

Alphabet:

- propositional letters: $a, b, c, \ldots$
- connectives: $\otimes, \multimapinv, \multimap$

Formulas: usual definition
Sequents: $\Gamma \vdash A$ where Γ is a finite sequence of formulas and A is a formula
Rules of sequent calculus:

$$\frac{}{A \vdash A}\ (axiom) \qquad \frac{\Gamma \vdash A \quad \Gamma_1, A, \Gamma_2 \vdash B}{\Gamma_1, \Gamma, \Gamma_2 \vdash B}\ (cut)$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ (r - \otimes) \qquad \frac{\Gamma_1, A, B, \Gamma_2 \vdash C}{\Gamma_1, A \otimes B, \Gamma_2 \vdash C}\ (l - \otimes)$$

$$\frac{\Gamma_1, A, \Gamma_2 \vdash C \quad \Delta \vdash B}{\Gamma_1, A \multimapinv B, \Delta, \Gamma_2 \vdash C}\ (l - \multimapinv) \qquad \frac{\Gamma, B \vdash A}{\Gamma \vdash A \multimapinv B}\ (r - \multimapinv)$$

$$\frac{\Gamma_1, A, \Gamma_2 \vdash C \quad \Delta \vdash B}{\Gamma_1, \Delta, B \multimap A, \Gamma_2 \vdash C}\ (l - \multimap) \qquad \frac{B, \Gamma \vdash A}{\Gamma \vdash B \multimap A}\ (r - \multimap)$$

Fig. 6. Language and sequent calculus for Multiplicative Noncommutative Intuitionistic Linear Logic

to two subparts of the tree where adjunction is done. However, it remains to prove that there is only one way to combine the pieces, the substitution node being given, and that the order of the elements are as requested. For that purpose, we show that for a suitable fragment of N-ILL there is a unique way to decompose a sequent $\Gamma, a \multimapinv A, \Delta \vdash B$ into $\Gamma, a, \Delta_2 \vdash B$ and $\Delta_1 \vdash A$. In this section, we clarify the calculus $\mathcal{A}$ used to interpret TAG: it includes a cut rule and an adjunction rule that mimic the grammatical operations. According with the previous remarks, these two rules are correct w.r.t. the logic. We give the basic properties satisfied by this calculus $\mathcal{A}$. In order to model TAG in N-ILL, we first construct the set $\mathcal{G}$ of subtrees of depth 1 of trees appearing in a TAG grammar $\mathcal{G}'$. The TAG grammar $\mathcal{G}'$ is then a subset of the closure $\mathcal{T}(\mathcal{G})$ under substitution (possibly with the declaration of adjunction nodes annotated in this case subst*) and adjunction of the set $\mathcal{G}$. The interpretation of elements of $\mathcal{G}$ as provable sequents of $\mathcal{A}$ is straightforward. This leads to a calculus $\mathcal{A}(\mathcal{G})$ where the operations are restricted w.r.t. $\mathcal{G}$. The TAG grammar $\mathcal{G}'$ is then in correspondence with a subset $\mathrm{M}(\mathcal{G}')$ of $\mathcal{A}(\mathcal{G})$ and we prove the equivalence between the language generated by $\mathcal{G}'$ and the set of sequents obtained by closure on $\mathrm{M}(\mathcal{G}')$ by the cut and adjunction rules[5]. Proofs of propositions are postponed until the annex. The various components of our approach are summarized below.

[5] We note M instead of $\mathrm{M}(\mathcal{G}')$ whenever there is no ambiguity.

$$\dfrac{\dfrac{\dfrac{}{NP \circ\!\!- \mathit{John}, \mathit{John} \vdash NP}\text{(lex)} \quad \dfrac{\dfrac{\dfrac{}{NP, (((NP \multimap S) \circ\!\!- NP) \circ\!\!- NP) \circ\!\!- \mathit{gives}, \mathit{gives}, NP, NP \vdash S}\text{(lex)} \quad \dfrac{}{NP \circ\!\!- \mathit{Mary}, \mathit{Mary} \vdash NP}\text{(lex)}}{NP, (((NP \multimap S) \circ\!\!- NP) \circ\!\!- NP) \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, NP \vdash S}\text{(cut)} \quad \dfrac{\dfrac{}{(NP \circ\!\!- N) \circ\!\!- a, a, N \vdash NP}\text{(lex)} \quad \dfrac{}{N \circ\!\!- \mathit{book}, \mathit{book} \vdash N}\text{(lex)}}{(NP \circ\!\!- N) \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash NP}}{NP, (((NP \multimap S) \circ\!\!- NP) \circ\!\!- NP) \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, (NP \circ\!\!- N) \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash S}\text{(cut)}}{NP \circ\!\!- \mathit{John}, \mathit{John}, (((NP \multimap S) \circ\!\!- NP) \circ\!\!- NP) \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, (NP \circ\!\!- N) \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash S}\text{(cut)}$$

Fig. 7. Proof of *John gives Mary a book*: (Lambek-style) two implications

$$\dfrac{\dfrac{}{NP \circ\!\!- \mathit{John}. \mathit{John} \vdash NP}\text{(lex)} \quad \dfrac{\dfrac{\dfrac{}{S \circ\!\!- NP \otimes VP. NP, VP \circ\!\!- V \otimes NP \otimes NP, V \circ\!\!- \mathit{gives}, \mathit{gives}, NP, NP \vdash S}\text{(lex)} \quad \dfrac{}{NP \circ\!\!- \mathit{Mary}, \mathit{Mary} \vdash NP}\text{(lex)}}{S \circ\!\!- NP \otimes VP, NP, VP \circ\!\!- V \otimes NP \otimes NP, V \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, NP \vdash S}\text{(cut)} \quad \dfrac{\dfrac{}{NP \circ\!\!- Det \otimes N, Det \circ\!\!- a, a, N \vdash NP}\text{(lex)} \quad \dfrac{}{N \circ\!\!- \mathit{book}, \mathit{book} \vdash N}\text{(lex)}}{NP \circ\!\!- Det \otimes N, Det \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash NP}\text{(cut)}}{S \circ\!\!- NP \otimes VP, NP, VP \circ\!\!- V \otimes NP \otimes NP, V \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, NP \circ\!\!- Det \otimes N, Det \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash S}\text{(cut)}}{S \circ\!\!- NP \otimes VP, NP \circ\!\!- \mathit{John}, \mathit{John}, VP \circ\!\!- V \otimes NP \otimes NP, V \circ\!\!- \mathit{gives}, \mathit{gives}, NP \circ\!\!- \mathit{Mary}, \mathit{Mary}, NP \circ\!\!- Det \otimes N, Det \circ\!\!- a, a, N \circ\!\!- \mathit{book}, \mathit{book} \vdash S}\text{(cut)}$$

Fig. 8. Proof of *John gives Mary a book*: one implication and times

$$\begin{tikzcd}
 & \mathcal{A}(\mathcal{G}) \arrow[loop above, "\text{cut}\\ \text{adj}"] \arrow[dd, shift right, "\text{Tree}"'] & \supset & \mathrm{M}(\mathcal{G}') \arrow[r, "\text{cut}\\ \text{adj}"] & \mathrm{CL}(\mathrm{M}(\mathcal{G}')) \\
\mathcal{G} \arrow[ur, "\circ\!-_{\mathcal{G}}\\ \otimes"] \arrow[dr, "\text{adj}\\ \text{subst}^{*}"'] & & & & \\
 & \mathcal{T}(\mathcal{G}) \arrow[uu, shift right, "\text{Seq}"'] & \supset & \mathcal{G}' \arrow[uu] \arrow[r, "\text{subst}\\ \text{adj}"'] & \mathrm{CL}(\mathcal{G}') \arrow[uu]
\end{tikzcd}$$

Consider the following fragment $\mathcal{A}$ of the non-commutative intuistionistic linear logic (N-ILL).

Definition 2 (The calculus $\mathcal{A}$).

- Alphabet of $\mathcal{A}$: propositional letters $a, b, \ldots$, connectives $\otimes, \circ\!-$,
- Formulas: usual definition. A is a *simple $\otimes$–formula* iff A is a propositional letter or A is a formula $b_1 \otimes \cdots \otimes b_n$ where $b_1, \ldots, b_n$ are propositional letters.
- Sequents: $\Gamma \vdash A$, where Γ is a finite sequence of formulas and A is a formula,
- Sequent calculus:
  - Axiom: $a \vdash a$
  - Rules: $\dfrac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,(\otimes) \qquad \dfrac{\Gamma \vdash A \quad \Gamma_1, a, \Gamma_2 \vdash B}{\Gamma_1, a \circ\!- A, \Gamma, \Gamma_2 \vdash B}\,(\circ\!-)$

Axiom and rules are restricted as follows: a stands for a propositional letter, A, B stand for simple $\otimes$–formulas.

Proposition 3 (Main properties of calculus $\mathcal{A}$). *(proofs in [5])*

1. *If $\Gamma \vdash A \otimes B$ is provable in $\mathcal{A}$, then*
   - *A and B are simple $\otimes$–formulas;*
   - *there is a unique pair (Γ_1, Γ_2) s.t. $\Gamma = \Gamma_1, \Gamma_2$ and both the sequents $\Gamma_1 \vdash A$ and $\Gamma_2 \vdash B$ are provable in $\mathcal{A}$.*
2. *If $\Gamma, a \circ\!- A, \Delta \vdash B$ is provable in $\mathcal{A}$, then*
   - *A and B are simple $\otimes$–formulas;*
   - *there is a unique pair (Δ_1, Δ_2) s.t. $\Delta = \Delta_1, \Delta_2$ and both the sequents $\Delta_1 \vdash A$ and $\Gamma, a, \Delta_2 \vdash B$ are provable in $\mathcal{A}$.*

   Such a pair (Δ_1, Δ_2) will be called "the splitting pair for Δ in $\Gamma, a \circ\!- A, \Delta \vdash B$". Note that this pair can be computed easily: the first element Δ_1 of the splitting pair must satisfy a counting condition on each variable occurring in it (see [5]).

3. The calculus $\mathcal{A}$ is closed under the atomic cut-rule *(i.e. substitution rule)*

$$\frac{\Gamma \vdash a \quad \Delta_1, a, \Delta_2 \vdash A}{\Delta_1, \Gamma, \Delta_2 \vdash A}\ (cut)$$

i.e.: if the sequents $\Gamma \vdash a$ and $\Delta_1, a, \Delta_2 \vdash A$ are provable in $\mathcal{A}$, then the sequent $\Delta_1, \Gamma, \Delta_2 \vdash A$ is also provable in $\mathcal{A}$.

4. The calculus $\mathcal{A}$ is closed under the adjoining rule

$$\frac{\Gamma_1, a, \Gamma_2 \vdash a \quad \Delta, a \multimapdotinv a, \Lambda \vdash b}{\Delta, \Gamma_1, \Lambda_1, \Gamma_2, \Lambda_2 \vdash b}\ (adj)$$

where (Λ_1, Λ_2) is the splitting pair of Λ in $\Delta, a \multimapdotinv a, \Lambda \vdash b$.

Note that Λ_1 and Λ_2 are *uniquely defined* from the premises, so the previous deduction is really a logical rule.

Definition 4 (The calculus $\mathcal{A}(\mathcal{G})$). Let $\mathcal{G}$ be a family of labeled trees, of depth 1, not of the form $X \multimapdotinv X$. Let $\mathcal{T}(\mathcal{G})$ be the closure of $\mathcal{G}$ under the rules:

- substitution with or without the declaration of a new internal point on which the adjoining operation may be applied,
- adjoining operation.

$\mathcal{A}(\mathcal{G})$ is the calculus obtained from $\mathcal{A}$ as follows:

- propositional letters are exactly all the labels of the trees in $\mathcal{G}$,
- the rule ($\multimapdotinv$) is restricted as follows:

$$\frac{\Gamma \vdash A \quad \Gamma_1, a, \Gamma_2 \vdash B}{\Gamma_1, a \multimapdotinv A, \Gamma, \Gamma_2 \vdash B}\ (\multimapdotinv, \mathcal{G})$$

  where A, B are simple $\otimes$–formulas of the language of $\mathcal{A}(\mathcal{G})$, a is a propositional letter of the language of $\mathcal{A}(\mathcal{G})$ and one of the following cases occurs:
  - A is a
  - A is a propositional letter b different from a, and the tree $\begin{tikzcd}[row sep=small] a \arrow[d, dash] \\ b \end{tikzcd} \in \mathcal{G}$
  - A is $b_1 \otimes \cdots \otimes b_n$, and the tree $\begin{tikzcd}[row sep=small, column sep=small] & a \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ b_1 & \dots & b_n \end{tikzcd} \in \mathcal{G}$

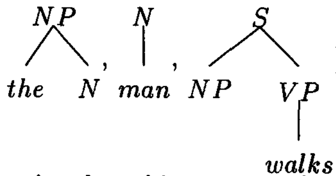
The following propositions state the correspondence between sequents and trees. The first two provide a precise translation between the two notions. Basically, a sequent $\Gamma \vdash a$ (in the previous language) is the logical equivalent of a tree with root a, and there is exactly one formula in Γ for each leaf, for each subtree (of depth 1), for each adjunction node, and nothing else. *Seq()* (resp. *Tree()*) associates a sequent to each tree (resp. each sequent), and we prove the two are converse. The last three propositions are properties concerning the logical counterpart of a TAG grammar. The last one is in fact the most important: the closure under (logical) adjunction and substitution of the set of sequents corresponding to a set of elementary trees is exactly the set of sequents corresponding to the closure under (grammatical) adjunction and substitution of this set of elementary trees. In other words, the logical and grammatical calculi coincide, i.e. the restricted logical calculus we defined above and the TAG calculus.

Proposition 5 (Main properties of calculus $\mathcal{A}(\mathcal{G})$). *The properties 1-4 of $\mathcal{A}$ are also properties of $\mathcal{A}(\mathcal{G})$. Moreover the following properties hold for $\mathcal{A}(\mathcal{G})$:*

- *To $T \in \mathcal{T}(\mathcal{G})$, we associate a sequent Seq(T) of $\mathcal{A}(\mathcal{G})$ s.t.:*
  - *if a is the root of T, and the terminal points of T (ordered from left to right) are $a_1, \ldots, a_m$, then Seq(T) is*
    $$\Gamma \vdash a$$
    where in Γ the sequence of all the occurring propositional variables is $a_1, \ldots, a_m$ and in Γ there is a formula $c \multimapinv c$ iff c is an internal point of T on which the adjoining operation may be operated;
  - *Seq(T) is provable in $\mathcal{A}(\mathcal{G})$.*
- *To every provable sequent $\Gamma \vdash A$ in $\mathcal{A}(\mathcal{G})$, we associate Tree($\Gamma \vdash A$) s.t.*
  - *if A is a propositional letter, then Tree($\Gamma \vdash A$) $\in \mathcal{T}(\mathcal{G})$ where the root is A, the terminal points (from left to right) are exactly all the propositional letters occurring in Γ and in the same order in which they occur in Γ, and the internal points on which the adjoining operation may be operated are exactly all the propositional letters c s.t. $c \multimapinv c$ occurs in Γ;*
  - *if A is $b_1 \otimes \cdots \otimes b_n$, and so $\Gamma = \Gamma_1 \ldots \Gamma_n$ with the sequents $\Gamma_i \vdash b_i$ provable in $\mathcal{A}(\mathcal{G})$ for every $1 \leq i \leq n$, then Tree($\Gamma \vdash A$) is a sequence $T_1, \ldots, T_n$ of trees $\in \mathcal{T}(\mathcal{G})$, s.t. $T_i =$ Tree($\Gamma_i \vdash b_i$).*
- *If $\Gamma \vdash a$ is provable in $\mathcal{A}(\mathcal{G})$, then Seq(Tree($\Gamma \vdash a$)) $= \Gamma \vdash a$. If T is a tree of $\mathcal{G}$, then Tree(Seq(T)) $= T$.*
- *Let M be a set of provable sequents in $\mathcal{A}(\mathcal{G})$. Define CL(M) as follows:*
  - *$M \subseteq$ CL(M)*
  - *(closure under atomic cut-rule)*
    if $\Gamma \vdash a \in$ CL(M) and $\Delta_1, a, \Delta_2 \vdash B \in$CL(M), then $\Delta_1, \Gamma, \Delta_2 \vdash B \in$CL(M)
  - *(closure under adjoining operation) if $\Gamma_1, a, \Gamma_2 \vdash a \in$CL(M) and $\Delta, a \multimapinv a, \Lambda \vdash b \in$CL(M), then $\Delta, \Gamma_1, \Lambda_1, \Gamma_2, \Lambda_2 \vdash b \in$CL(M), where (Λ_1, Λ_2) is the splitting pair of Λ in $\Delta, a, \Lambda \vdash b$;*
  - *nothing else belongs to CL(M).*
- *If $\Gamma \vdash A \in$CL(M), then $\Gamma \vdash A$ is provable in $\mathcal{A}(\mathcal{G})$.*
- *If $\mathcal{G}' \subseteq \mathcal{T}(\mathcal{G})$, let CL($\mathcal{G}'$) be the closure of $\mathcal{G}'$ under:*
  - *substitution;*
  - *adjoining operation.*

  Clearly, CL($\mathcal{G}'$) $\subseteq \mathcal{T}(\mathcal{G})$.
  Let $M = \{Seq(T)/T \in \mathcal{G}'\}$, then $CL(M) = \{Seq(T)/T \in CL(\mathcal{G}')\}$.

Starting from this last proposition, it is not too difficult to prove that the language accepted by a TAG grammar $\mathcal{G}'$ is exactly the language accepted by M($\mathcal{G}'$). We can define the language accepted by such a calculus in the following way. Let us take only those sequents in CL(M($\mathcal{G}'$)) whose right part is the propositional variable S (the start symbol of the grammar), and such that propositional variables of the left part of the sequent correspond to terminal symbols of the grammar, i.e. words of the language. The language accepted by M($\mathcal{G}'$) is then the set of sequences of words in the same order as they appear in the previous sequents.

Example 6. Grammar $\mathcal{G}_1' = \{$  $\}$

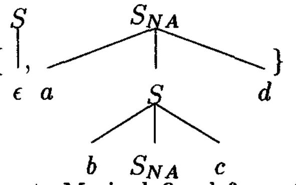
The set of sequents M_1 associated to this grammar is the following one:
$\mathrm{M}_1 = \{$

$NP \multimapinv the \otimes N, the, N \vdash NP,$
$N \multimapinv man, man \vdash N,$
$S \multimapinv NP \otimes VP, NP, VP \multimapinv walks, walks \vdash S$

$\}$

The analysis of "the man walks" corresponds to the following proof in $\mathcal{A}(\mathcal{G}'_1)$:

$$\frac{\dfrac{NP \multimapinv the \otimes N, the, N \vdash NP \quad N \multimapinv man, man \vdash N}{NP \multimapinv the \otimes N, the, N \multimapinv man, man \vdash NP} \qquad S \multimapinv NP \otimes VP, NP, VP \multimapinv walks, walks \vdash S}{S \multimapinv NP \otimes VP, NP \multimapinv the \otimes N, the, N \multimapinv man, man, VP \multimapinv walks, walks \vdash S}$$

Example 7. Grammar $\mathcal{G}_2' = \{$  $\}$

The associated set of sequents M_2 is defined from $\mathcal{G}_2'$
$\mathrm{M}_2 = \{$

$S \multimapinv a \otimes S \otimes d, a, S \multimapinv S, S \multimapinv b \otimes S \otimes c, b, S, c, d \vdash S,$
$S \multimapinv S, S \multimapinv \epsilon, \epsilon \vdash S$

$\}$

The analysis of "aabbccdd" corresponds to the proof of the following sequent. We have decomposed the different elements of the left part according to the adjunction rule.

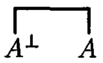
$$\underbrace{S \multimapinv a \otimes S \otimes d, a,}_{\Delta} \underbrace{S \multimapinv a \otimes S \otimes d, a, S \multimapinv S, S \multimapinv b \otimes S \otimes c, b,}_{\Gamma_1} \underbrace{S \multimapinv b \otimes S \otimes c, b, S, c,}_{\Lambda_1} \underbrace{c, d,}_{\Gamma_2} \underbrace{d}_{\Lambda_2} \vdash S$$

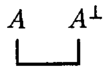
5 TAG analysis using noncommutative proof nets

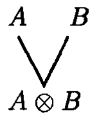
A proof in sequent calculus contains many useless properties in its contexts. Girard has defined in a purely geometric way [6] a class of graphs of formulas, called *proof-nets*: to each proof of a sequent $\vdash \Gamma$ in the one-sided sequent calculus for multiplicative linear logic corresponds a proof-net whose conclusions are exactly the formulas in Γ, and to each proof-net corresponds at least one proof of the sequent $\vdash \Gamma$ in the one-sided sequent calculus for multiplicative linear logic (where Γ is a sequence of all the conclusions of the proof-net). Similarly, Abrusci [3] defined in a purely geometric way a class of graphs, called *noncommutative proof nets*: to each proof of a sequent $\vdash \Gamma$ in the one-sided sequent calculus for multiplicative noncommutative linear logic corresponds a noncommutative proof-net with conclusions Γ, and to each noncommutative proof-net with conclusions Γ corresponds at least one proof of the sequent $\vdash \Gamma$ in the one-sided sequent calculus for multiplicative noncommutative linear logic. Therefore, to each proof of $\Gamma \vdash A$ in the sequent calculus for intuistionistic multiplicative noncommutative linear logic corresponds a noncommutative proof-net with conclusions $(\Gamma^*)^\perp, A^*$.

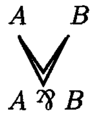
5.1 Noncommutative proofnets

To every proof π of a sequent $\vdash \Gamma$ in the one-sided sequent calculus for multiplicative noncommutative linear logic, we can associate (by induction on the construction of the proof π) a *noncommutative proofnet with conclusions* Γ, i.e. an oriented planar graph π' of occurrencies of formulas s.t.:

- the conclusions of π' are exactly the formulas in Γ;
- π' is a noncommutative proof structure, i.e. it is constructed by means of the following links[6]:
  - Axiom-link (two conclusions, no premise) 

  - Cut-link (two premises, no conclusion) 

  - $\otimes$-link (two premises, one conclusion) 

  - $⅋$-link (two premises, one conclusion) 

  and every occurrence of formula is a premise of at most one link and is conclusion of exactly one link;
- the translation of π is a proofnet, i.e. it admits no shorttrip. A shorttrip is a trip that does not contain each node twice. A trip is a sequence of nodes, going from one node to another according to the graph and to a switch for each times-link and each par-link, in a bideterministic way: the traversal of nodes is done according to fig. 9 but without taking into account the labels of nodes (see below);
- every assignment for π' is total;
- π' induces the linear order Γ of the conclusions.

An *assignment* for a proof structure π' is made in the following way. Let us associate two integer variables left-N and right-N to each node N computed as in fig. 9. The left variable of a node labeled by a propositional variable A is named

[6] The par link is graphically distinguished from the times link. However this is only sugar as the graph has really only one kind of edge.

x^A. A *special trip* is a trip from the left variable to the right variable of some node. It follows the links given by the graph but the switch for times must be at right, the switch for par must be at left. Moreover the switch for par is used only if right-C occurs before right-B in the special trip from left-A. At the same time it imposes constraints between integer variables as defined in the following figure, where $\dashrightarrow$ means the transition from a variable to another variable. The assignment is *total* when the set of constraints can be satisfied. The precedence relation on the conclusions of π' is defined s.t. A precedes B iff right-A = left-B + 1. Then π' *induces the linear order of the conclusions* iff the precedence relation is a chain and each conclusion occurs exactly once in the chain.

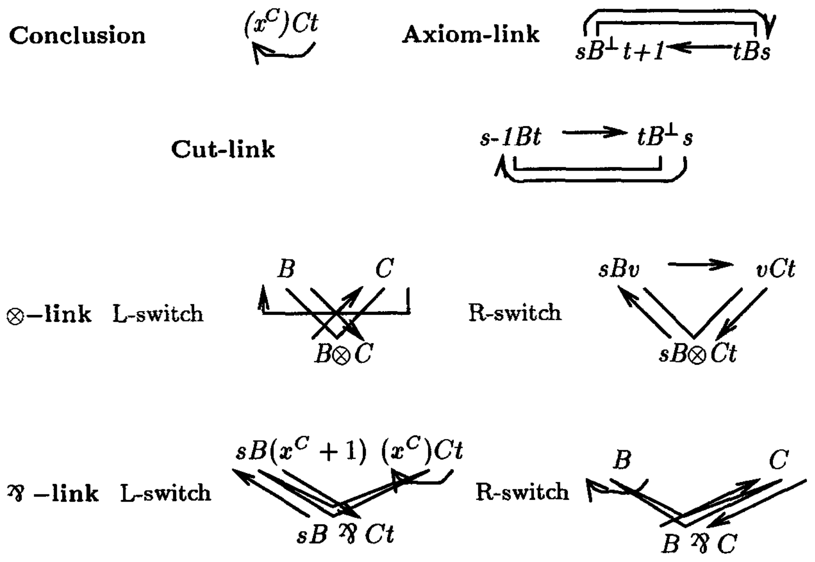


Fig. 9. Travels through proof structures

Precise definitions, examples, explanations and the proof of the following theorem can be found in [4].

Theorem 8. *π' is a noncommutative proof net with conclusions Γ iff there exists a proof π of the sequent $\vdash \Gamma$ in the sequent calculus for multiplicative noncommutative linear logic s.t. π' is associated to π.*

Note that every noncommutative proofnet is a planar graph.

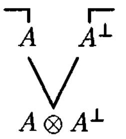
5.2 Parsing examples

We give in this section two simple examples of parses. The aim of this section is to show the strong connection between the structure of proofs of sequents and a standard TAG derived structure. Moreover, it emphasizes the interest of a proofnet approach as the syntax (and parsing process) is concretely designed as a logical manipulation of logical structures. In the conclusion, we briefly mention that this can also give a logical formalization of D-trees [13].

The first example requires only composition, i.e. the cut-rule from the logical point of view. We first give the sequents (provable in $\mathcal{A}$) associated with the lexical items. Their meanings are straightforward, e.g. '*John* and *Mary* are noun phrases (NP)', '*saw* requires a complement NP to obtain a verb phrase (VP) and a subject NP to obtain a sentence (S)'. Note that VP is an adjunction node so the sequent associated to the item *saw* includes the formula $VP \circ\!- VP$. The next example uses this specification.

John $NP \circ\!- John, John \vdash NP$
Mary $NP \circ\!- Mary, Mary \vdash NP$
saw $S \circ\!- NP \otimes VP, NP, VP \circ\!- VP, VP \circ\!- V \otimes NP, V \circ\!- saw, saw, NP \vdash S$

The proof associated to the analysis of *John saw Mary* requires two cuts. The two sequent proofs given in fig. 10 are the only two possibilities for this sentence in the fragment $\mathcal{A}(\mathcal{G})$. This pinpoints the fact that the order in which the cuts are made is not significant w.r.t. the derived structure. Proofnets allow expression of this equivalence. Hence the two proofs have the same associated proofnet given in fig. 11. For the sake of clarity, the cut rules are bold lines, and we circle subnets associated to lexical items. Obviously, if we delete the two cut lines, we are left with three proofnets representing (provable) sequents. Such a proofnet contains still superfluous information. As the only available operations in $\mathcal{A}(\mathcal{G})$ are (i) the cut-rule and (ii) the adjunction rule on a propositional variable, we only need to keep nodes referring to (i) conclusions of the proofnet that are propositional variables or negation of propositional variables (a cut can be made on such a literal), and (ii) the fact that there exists a subgraph of the following form (corresponding to the existence of a formula $A \circ\!- A$ in the left part of a sequent, i.e. its negation $A \otimes A^{\perp}$ in the one-sided associated sequent):



We can then simplify the graph and replace the internal logical machinery by black boxes (big black circles in the figure). The conclusions of the basic proofnets are labeled: outputs (i.e. conclusions that are propositional variables) are drawn as closed half circles, inputs (i.e. conclusions that are negation of propositional variables) are drawn as open half circles. Plain lines link black boxes to black boxes or conclusions, and subgraphs corresponding to adjunction points are drawn as dashed lines. The previous proofnet is then redrawn as in fig. 14 right.

$$\dfrac{NP \multimapinv John, John \vdash NP \qquad \dfrac{S \multimapinv NP \otimes VP, NP, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \vdash S \qquad NP \multimapinv Mary, Mary \vdash NP}{S \multimapinv NP \otimes VP, NP, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \multimapinv Mary, Mary \vdash S}}{S \multimapinv NP \otimes VP, NP \multimapinv John, John, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \multimapinv Mary, Mary \vdash S}$$

$$\dfrac{NP \multimapinv Mary, Mary \vdash NP \qquad \dfrac{S \multimapinv NP \otimes VP, NP, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \vdash S \qquad NP \multimapinv John, John \vdash NP}{S \multimapinv NP \otimes VP, NP \multimapinv John, John, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \vdash S}}{S \multimapinv NP \otimes VP, NP \multimapinv John, John, VP \multimapinv VP, VP \multimapinv V \otimes NP, V \multimapinv saw, saw, NP \multimapinv Mary, Mary \vdash S}$$

Fig. 10. $\mathcal{A}(\mathcal{G})$-proofs of *John saw Mary*

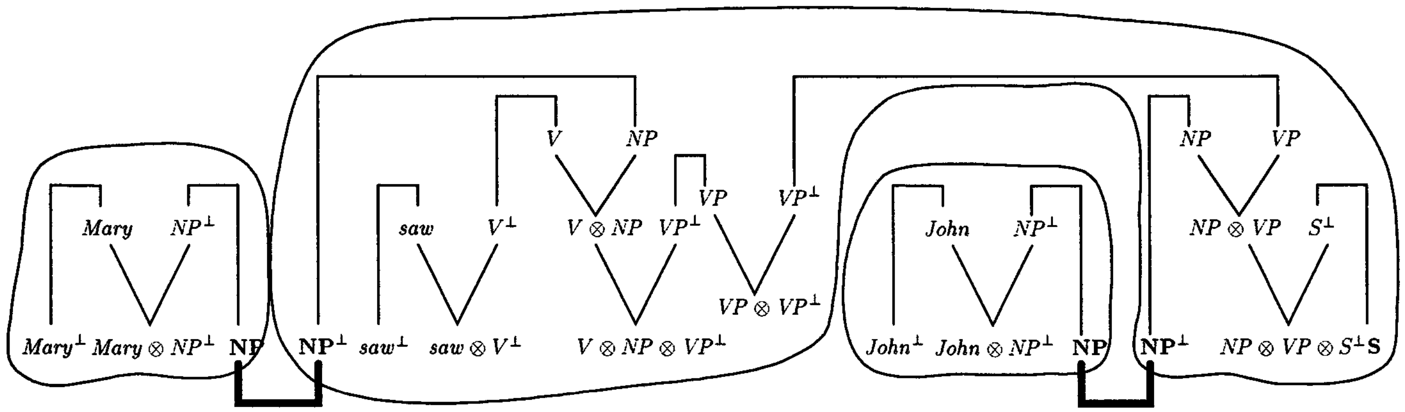


Fig. 11. *John saw Mary*

We obviously find the derived tree (neglecting some minor differences). The logical proofnet can then be seen as an "explanation" of the structure of the tree, that is to say the operations available on the tree are the result of some focus of what can be done on the proofnet. On the one hand the use of black boxes is necessary to clarify the structure of the analysis, on the other hand this hides proof details that can be useful for some linguistic operations (as is the case for adjunction w.r.t. the classical structure of a derived tree). We show in [5] another application of such a (logical) refinement.

The next example includes the two operations composition and adjunction, i.e. two cut-rules and an adjunction rule. In fig. 12 the adjunction rule is represented as a double thick dashed line: this (logically) mimics the adjunction as it can be described in the derived tree given in fig. 14 left. Note that the adverb has to be placed after the complement (rightmost in the proofnet) in order to keep the graph planar. The proofnet in fig. 13 is the proofnet corresponding to a cut-free proof. The sequent associated to the adverb *today* is the following one:
$\textit{today}\ VP \multimap VP \otimes \textit{today}, VP, \textit{today} \vdash VP$

6 Conclusion

The use of logics to describe Natural Language is not a new idea. Work on e.g. Lambek calculus and logic programming are famous examples (see e.g. [11], ... on how to use sequent calculus for natural language processing). However, linguistic formalisms have fundamentally evolved these two decades. Though theoretical research has been done on unification and attribute-value structures, operations on syntactic trees have been investigated mainly by comparing different solutions [16,15]. We consider here another way to look at these operations (see also [7]). We focus on the adjunction operation available in Tree Adjoining Grammars as it seems to be the most simple way to augment the expressive power of a formalism. We prove that Noncommutative Intuitionistic Linear Logic is a natural logical means and we define a fragment equivalent to TAG. We show furthermore to which extent geometric representations of proofs (proofnets) may be useful to understand how black boxes (i.e. relations between nodes in a syntactic tree) help simplifying a parse but also hide interesting mechanisms. There is still a lot to do in this direction. Among other things, Generalized Categorial Grammars have also to be logically investigated, the objective being to relate the current available operations and to complete this set. The previous discussions show also the relationship between our point of view and the idea of quasi-trees developed by Vijay-Shanker [13]. He proposes to consider *partial descriptions of trees*, i.e. adjunction nodes are represented by means of loose relations whose meaning is a domination relation. In this case, the adjunction operation is identified with a pair of substitution operations. The strong relation with what precedes is clear. However, in order to take into account exactly this presentation the axiom of identity $A \vdash A$, where A is a propositional variable, has to be added to the

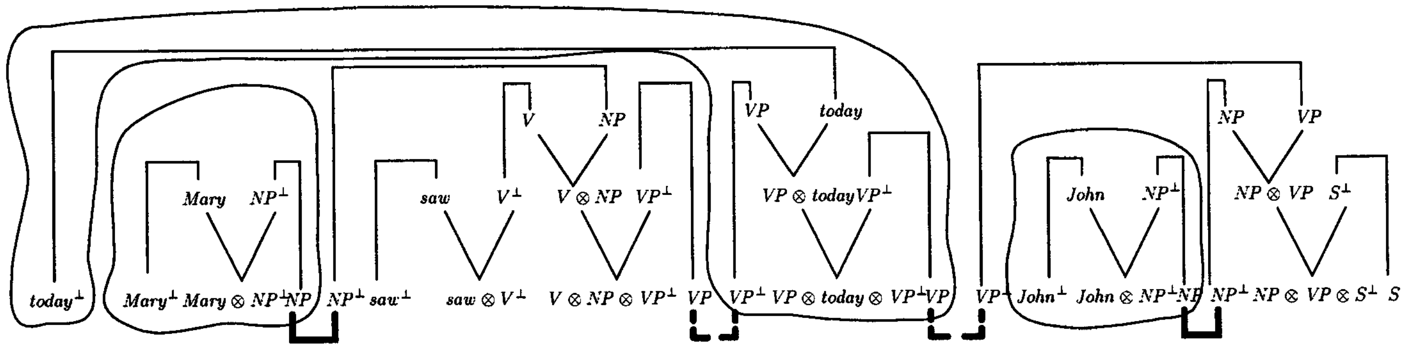

today⊥
Mary⊥ Mary ⊗ NP⊥ NP NP⊥ saw⊥ saw ⊗ V⊥ V ⊗ NP ⊗ VP⊥ VP VP⊥ VP ⊗ today ⊗ VP⊥ VP VP⊥ John⊥ John ⊗ NP⊥ NP NP⊥ NP ⊗ VP ⊗ S⊥ S
Mary NP⊥ saw V⊥ V ⊗ NP VP⊥ VP ⊗ today VP⊥ John NP⊥ NP ⊗ VP S⊥
V NP VP today NP VP

Fig. 12. *John saw Mary today*

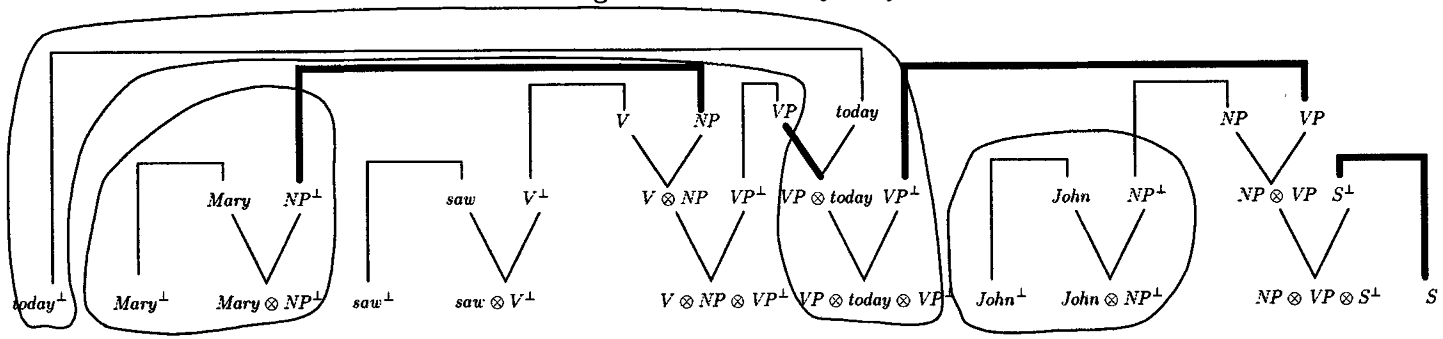
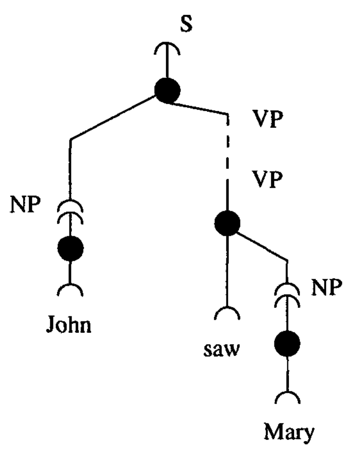

today⊥
Mary⊥ Mary ⊗ NP⊥ saw⊥ saw ⊗ V⊥ V ⊗ NP ⊗ VP⊥ VP ⊗ today ⊗ VP⊥ John⊥ John ⊗ NP⊥ NP ⊗ VP ⊗ S⊥ S
Mary NP⊥ saw V⊥ V ⊗ NP VP⊥ VP ⊗ today VP⊥ John NP⊥ NP ⊗ VP S⊥
V NP VP today NP VP

Fig. 13. Cut-free proofnet for *John saw Mary today*



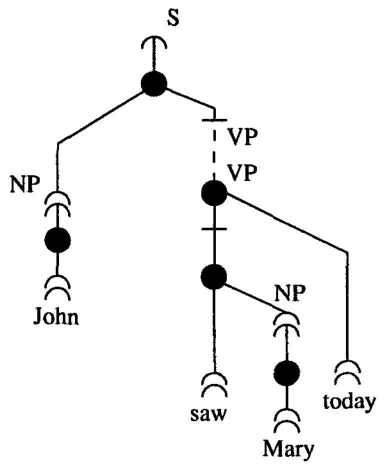


Fig. 14. (left) Simplified proof for *John saw Mary*. (right) Simplified proof for *John saw Mary today*

calculus $\mathcal{A}(\mathcal{G})$ given in section 4. In this way, adjunction nodes can be deleted from sequents. In this new calculus, the following rule is satisfied:

$$\frac{A \vdash A \quad \Gamma, A \multimapinv A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ (adjunction)$$

Hence, we obtain the following equivalence:

Proposition 9. *A parse tree is* correct

iff each pair of nodes in a domination relation have the same label
iff there is a proof whose conclusions that are propositional variables are the words of the sentence in the same order, and without any formula of the form $A \multimapinv A$.

References

1. A. Abeillé, K. Bishop, S. Cote, and Y. Schabes. A lexicalized tree-adjoining grammar for english. Technical Report MS-CIS-90-24, LINC LAB 170, Computer Science Department, University of Pennsylvania, Philadelphia, PA, 1990.
2. M. Abrusci. Noncommutative intuitionnistic linear propositional logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 36:297–318, 1990.
3. M. Abrusci. Phase semantics and sequent calculus for pure noncommutative classical linear propositional logic. *The Journal of Symbolic Logic*, 56(4):1403–1451, 1991.
4. M. Abrusci. Noncommutative proof nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, volume 222, pages 271–296. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, NewYork, June 1993.

5. M. Abrusci, C. Fouqueré, and J. Vauzeilles. Tree adjoining grammars in non-commutative linear logic. Technical Report 97-03, LIPN, Université Paris-Nord, France, 1997. In submission.
6. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
7. A.K. Joshi and S. Kulick. Partial proof trees, resource conscious logic and syntactic constraints. In *(this volume)*, 1997.
8. A.K. Joshi, L.S. Levy, and M. Takahashi. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163, 1975.
9. Aravind Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3 – Beyond Words, chapter 2, pages 69 – 124. Springer-Verlag, 1996.
10. A.S. Kroch and A.K. Joshi. Linguistic relevance of tree adjoining grammars. Technical Report MS-CIS-85-18, LINC LAB 170, Computer Science Department, University of Pennsylvania, Philadelphia, PA, 1985.
11. J. Lambek. The mathematics of sentence structure. *Am. Math. Monthly*, 65:154–169, 1958.
12. G. Sundholm. Systems of deduction. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 1, pages 133–188. D. Reidel Publishing Company, 1983.
13. K. Vijay-Shanker. Using descriptions of trees in a tree adjoining grammar. *Computational Linguistics*, 18(4):481–517, 1992.
14. K. Vijay-Shanker and A.K. Joshi. Some computational properties of tree adjoining grammars. In *23rd Meeting of the Association for Computational Linguistics*, pages 82–93, 1985.
15. K. Vijay-Shanker and D.J. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545, 1994.
16. K. Vijay-Shanker and D.J. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, 1994.